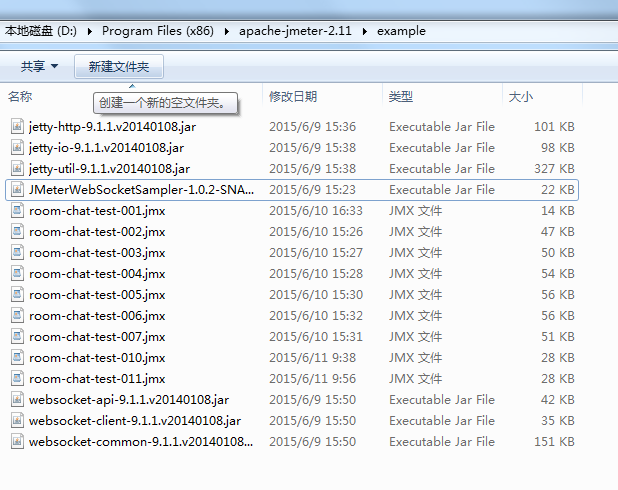
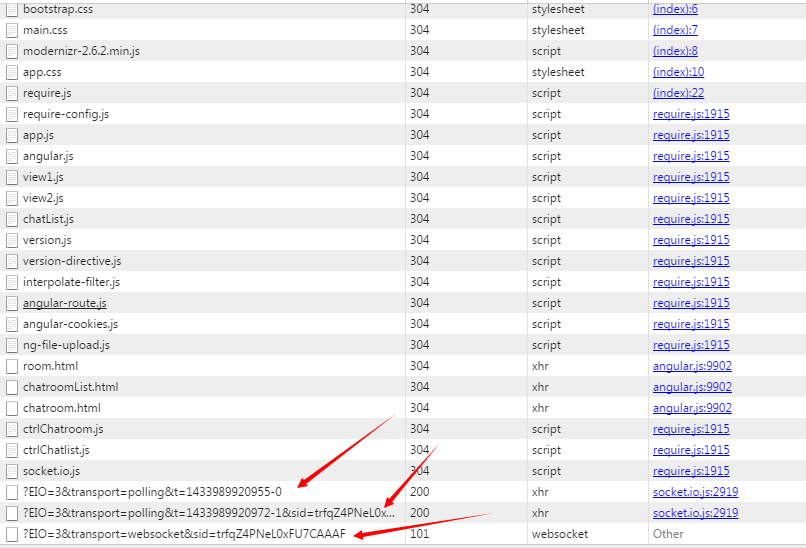
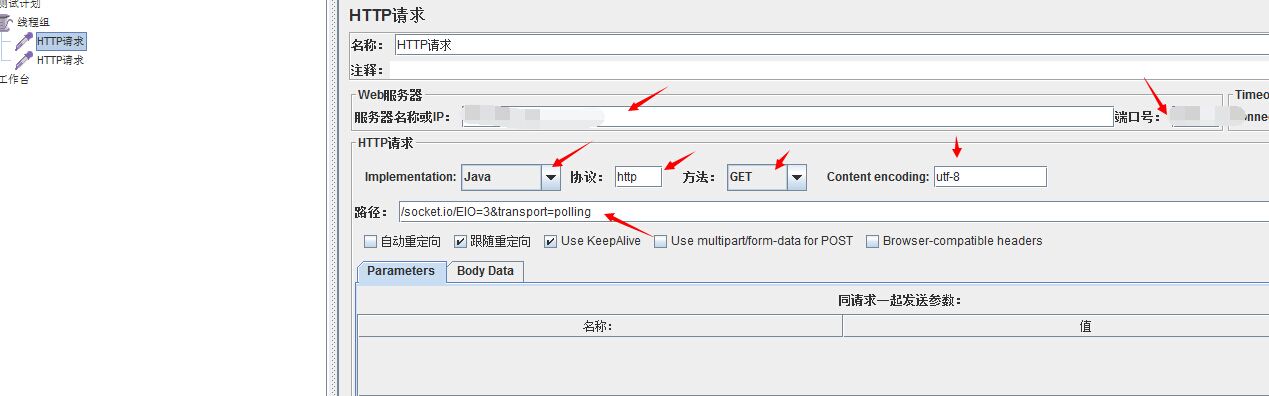
　　前段时间本着练习angularJS+requireJS的目的写了一个基于nodeJS和socket.io的聊天室，github地址为：<https://github.com/towersxu/node-socketIO-angular-require-chatroom>。当时没有考虑性能方面的问题，本着功能实现的目的就好。最近刚好有空，打算测试一下这个聊天室的性能。  
　　测试工具我使用的是JMeter，版本为2.11。这个是同事拷贝给我的绿色版本。只要有Java环境就可以运行（下载地址：apache-jmeter-2.11）。点击bin目录下的jmeter.bat就可以运行了。我这个版本本身是不支持websocket测试的，于是需要下载一个插件叫做JMeterWebSocketSampler-1.0.2-SNAPSHOT.jar，这个插件需要下载6个依赖包，如下图所示：



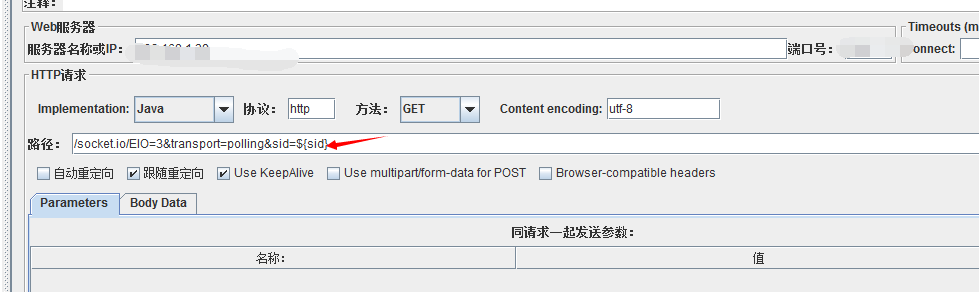
　　在插件安装好了后，就可以使用jmeter来模拟socket.io的通信了。首先打开chrome控制台，查看network。进入含有socket.io的页面时，我们可以看到如下图所示的几个链接。



　　我们只需要把这几个链接用jmeter实现了，就可以模拟浏览器和**[服务器](javascript:;" \t "_self)**的通信了。首先我们模拟前两个请求。点击请求我们可以看到这个请求的详细信息，从header上我们可以发现其实这两个请求还是HTTP请求，只不过带的参数不一样。第二个请求比第一个请求多带一个参数叫sid，他的值是第一个请求所相应的。这两个请求我们可以看做是TCP中的1次握手过程（我不懂TCP的三次握手，感觉这个应该和它差不多）或者说是一次认证，这个sid就是通信的唯一标识。下面我要使用jmeter模拟这两个请求。在线程组的sampler里面选择HTTP请求。如下图所示：



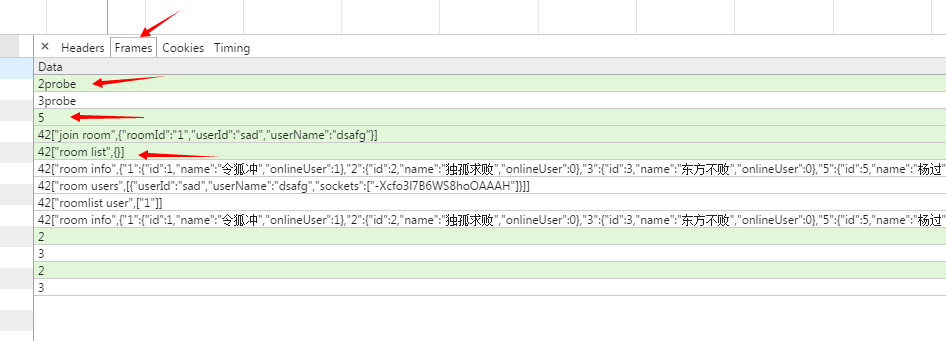
　　第一个请求（注意，上面图片路径应该为/socket.io/?EIO=3&transport=polling   少了个问号==!）



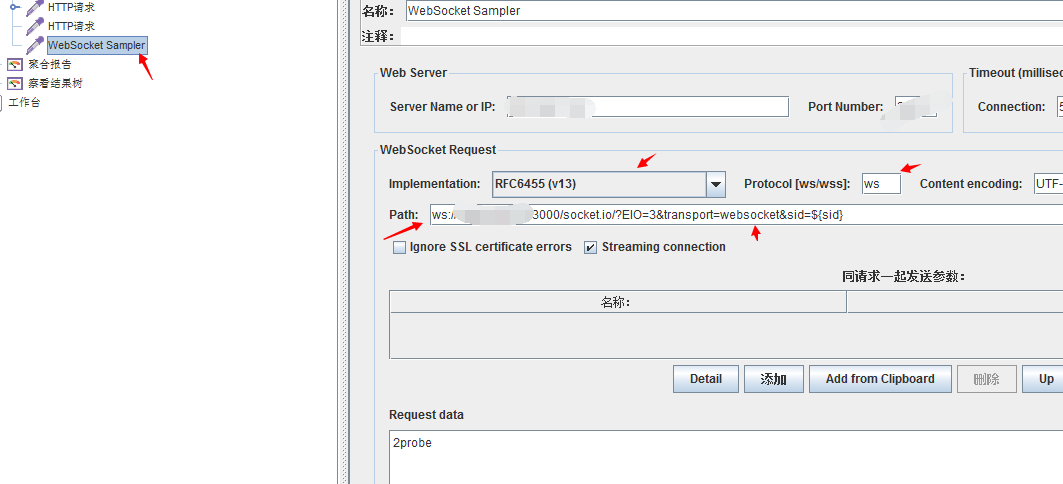
　　第二个请求注意，上面图片路径应该为/socket.io/?EIO=3&transport=polling   少了个问号==!  
　　注意第二个请求中的${sid}，这个是在第一个请求里面获取的，所有需要在第一个请求里面增加一个后置处理器-->正则表达式提取器。



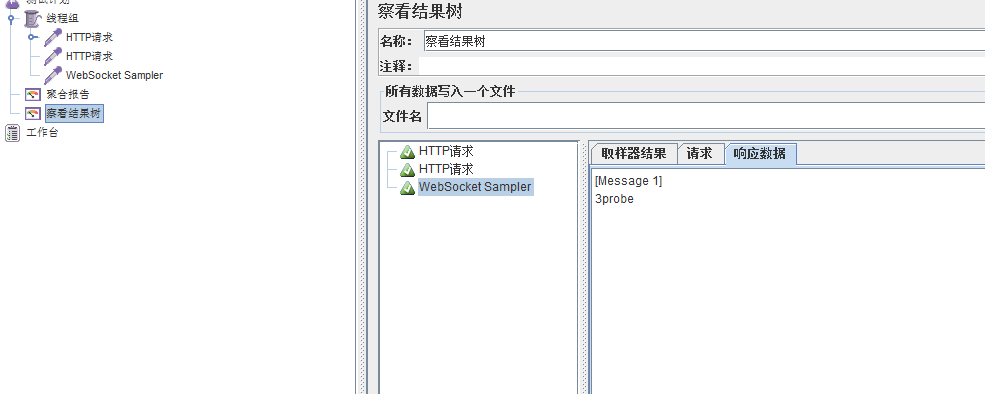
　　现在我到chrome里面查看第3个请求的详细信息，我们可以发现，第三个请求的URL变成了ws，参数transport也变成了websocket。并且多了一个frames栏目。Frames如下图所示：



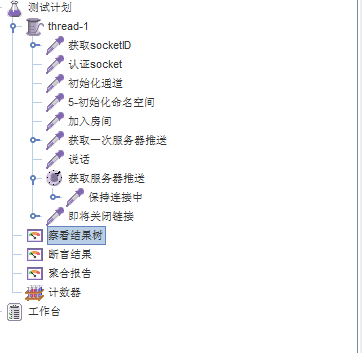
　　现在我们新建一个Websocket sampler来模拟socket链接。并发送2probe给服务器：



　　添加一个观察结果树，运行就可以看到请求的结果，相应数据为3probe



　　后面我们继续模拟socket.io的后续链接，也就是Frams里面的绿色部分，而白色部分就是服务器的响应。注意，最后要使用循环控制器间隔发送2给服务器保持连接。我的整个流程如下（上面下载的jmeter的example文件夹中的room-chat-test-010.jxm）：



　　当然最后的测试结果就不说了，同时加入300个用户就挂了。话说有没有人知道怎么模拟1秒钟加入100个人？我想测试加入5000人，分为50秒加入。